FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, DC 20005-3315

202 • 408 • 4000
FACSIMILE 202 • 408 • 4400

WRITER'S DIRECT DIAL NUMBER:

ATLANTA
404•653•6400
PALO ALTO
650•849•6600

TOKYO
011•813•3431•6943
BRUSSELS
011•322•646•0353

(202) 408-4148

February 16, 2000

ATTORNEY DOCKET NO. 06502-0210-00000

Box PATENT APPLICATION
Assistant Commissioner for Patents
Washington, D.C. 20231

Re: New U.S. Patent Application
Title: ADAPTIVE MEMORY ALLOCATION
Inventor(s): Paul HINKER, Bradley LEWIS
and Michael BOUCHER

Sir:

We enclose the following papers for filing in the United States Patent and Trademark Office in connection with the above patent application.

1. Application - 18 pages, including 5 independent claim(s) and 23 claims total.

2. Drawings - 5 sheets of informal drawings (Figures 1 - 4).

3. A check for $ 940.00 representing a $ 690.00 filing fee, $ 210.00 for additional claims and $40.00 for recording the Assignment.

Please accord this application a serial number and filing date and record and return the Assignment to the undersigned.

The Commissioner is hereby authorized to charge any additional filing fees due and any other fees due under 37 C.F.R. § 1.16 or § 1.17 during the pendency of this application to our Deposit Account No. 06-0916.

Respectfully submitted,

FINNEGAN, HENDERSON, FARABOW,
GARRETT & DUNNER, L.L.P.

By: _____
Lisa E. Marks
Reg. No. 44,901

LEM:kb
Enclosures

Sun Reference No.: P3689
Our Reference No.: 06502.0210

UNITED STATES PATENT APPLICATION

OF

Paul HINKER, Bradley LEWIS, and Michael BOUCHER

FOR

ADAPTIVE MEMORY ALLOCATION

FIELD OF THE INVENTION

This invention relates generally to data processing systems, and more particularly, to methods for optimizing the allocation and deallocation of shared memory to programs executing in a data processing system.

BACKGROUND OF THE INVENTION

During execution, programs typically make many dynamic memory access requests. Such requests involve requesting allocation of memory from a system call (e.g., malloc), utilizing the memory, and deallocating the memory using a system call (e.g., free). Dynamic memory allocation is a fundamental and very important part of many computer programs. It is therefore desirable to improve the performance of memory access functions.

SUMMARY OF THE INVENTION

In accordance with methods and systems consistent with the present invention, an improved memory access function (e.g., malloc) is provided that dynamically improves its performance by changing its operation at runtime. The memory access function adapts its operation based on previous memory access requests to be able to provide the fastest response to those kinds of memory access requests that it predicts will be the most common. For purposes of this description "access to system memory" includes both requests for allocation of system memory and release of system memory, or deallocation. Similarly, a "memory request" refers to either an allocation (a request for system memory), or a deallocation (a return of previously requested memory).

LAW OFFICES
FINNEGAN, HENDERSON,
FARABOW, GARRETT,
& DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

In accordance with methods and systems consistent with the present invention, as embodied and broadly described herein, a method is provided in a data processing system for allocating memory. The method receives a memory request for a reference to a block of memory. Then it returns the reference to the block of memory to satisfy the request. Next, it adjusts an operation of the memory access function based on the memory request.

Furthermore, in accordance with methods and system consistent with the present invention, as embodied and broadly described herein, a system for providing access to memory includes a memory which further includes a program including a memory access function that provides access to memory and adjusts its operation according to a memory request for a reference to a block of memory, and a processor for executing the program.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 depicts a block diagram of a data processing system suitable for practicing methods and systems consistent with the present invention;

Fig. 2 depicts in greater details the general access tree and the fast access tree depicted in Fig. 1;

Figs. 3A & 3B depict a flow diagram illustrating operations performed to allocate memory in accordance with the principles of the present invention; and

Fig. 4 depicts a flow diagram illustrating operations performed to deallocate memory in accordance with the principles of the present invention.

LAW OFFICES
FINNEGAN, HENDERSON,
FARABOW, GARRETT,
& DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

DETAILED DESCRIPTION

Methods and systems consistent with the present invention provide a function called "smart-alloc" that adapts in real-time the memory allocation process to threads executing in a data processing system. More specifically, smart-alloc observes past memory allocation behavior and adjusts the process of providing memory access such that the most frequently requested memory blocks of a specific size are allocated faster than less frequently requested memory block sizes. Smart-alloc is a language-independent library function which may be linked to one or more programs operating in a multi-processing environment.

Overview

When a program begins execution, smart-alloc secures access to an amount of memory that may be accessed as needed by threads executing within the program. Upon receiving a pointer from a system memory call to an amount of memory requested, smart-alloc divides the memory into blocks of varying sizes, grouping blocks of like size together in a linked-list, referred to as a free memory linked-list. Smart-alloc creates two trees that point to the free memory linked-lists: a fast access tree and a general access tree. As a program executes, the fast access tree points to the most frequently accessed memory block sizes, and the general access tree points to block sizes not pointed to by the fast access tree. Each free memory linked-list will be pointed to by either the general access tree, or the fast access tree, but not both.

Upon receiving a request from an executing thread for access to memory, smart-alloc increments a counter associated with the block size that will be used to satisfy the memory request. To satisfy the memory request, smart-alloc first determines whether the fast access tree

LAW OFFICES
FINNEGAN, HENDERSON,
FARABOW, GARRETT,
& DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

points to memory blocks of the size needed to satisfy the request. If the fast access tree includes memory blocks of the size needed to satisfy the request, smart-alloc satisfies the request from the fast access tree. Otherwise, smart-alloc satisfies the request from the general access tree. If a request is satisfied from the general access tree, smart-alloc compares the value of the counter associated with the allocated block size to the counter values for block sizes included in the fast access tree to determine whether the general access tree refers to block sizes most frequently requested. If the counter value for the allocated block size (from the general access tree) is greater than any of the counter values of block sizes included in the fast access tree, smart-alloc replaces that general access tree pointer to the block size having the highest counter value with a pointer from the fast access tree, and replaces the pointer to the block size included in the fast access tree having the lowest counter value with a pointer from the general access tree. This process ensures that the fast access tree continues to point to the most frequently requested block sizes.

Smart-alloc also receives requests from executing threads desiring to release references to blocks of memory. After receiving a released reference, smart-alloc adds the reference to the free memory linked-list that includes memory blocks corresponding to the size of the returned reference.

Implementation Details

Fig. 1 depicts a block diagram of a data processing system 100 suitable for practicing methods and implementing systems consistent with the present invention. Data processing system 100 includes computer 102 connected to network 105. Computer 102 includes secondary

LAW OFFICES
FINNEGAN, HENDERSON,
FARABOW, GARRETT,
& DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

storage 110, processors 120a...n, output device 130, input device 140, and memory 150. Memory 150 includes programs 155, 160, and 165, operating system 170, and shared memory 180. Operating system 170 includes a system shared memory access function, malloc 182. Each of programs 155, 160, and 165 includes smart-alloc 185, a shared memory access function operating in accordance with the principles of the present invention. Each instance of smart-alloc 185 includes two data structures, a fast access tree 186 and a general access tree 187 that are used by smart-alloc 185 when allocating and deallocating memory.

Programs 155, 160, and 165 share access to shared memory 180. Programs 155, 160, and 165 may include a single thread or multiple threads. Although multiple processors are shown, one skilled in the art will appreciate that programs 155, 160, and 165 may execute on a single processor to implement methods and practice systems consistent with the present invention.

Operating system 170 represents the Solaris® operating system from Sun Microsystems, Inc., which specifically facilitates multi-threaded program execution, although any operating system may be used to implement methods and systems consistent with the present invention.

Although aspects of this implementation are depicted as being stored in memory 150, one skilled in the art will appreciate that all or part of systems and methods consistent with the present invention may be stored on or read from other computer readable media, such as secondary storage devices, like hard disks, floppy disks, and CD-ROM; a digital signal received from a network such as the Internet; or other forms of RAM or ROM, either currently known or later developed. Sun, Sun Microsystems, and the Sun logo are trademark or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

LAW OFFICES
FINNEGAN, HENDERSON,
FARABOW, GARRETT,
& DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

Fig. 2 depicts general access tree 202 and fast access tree 204 pointing to free memory linked-lists 262-272. In this example, general access tree 202 includes two main branches: branch 206 including pointers to free memory linked-lists containing references to memory blocks of a size less than or equal to 4K and branch 208 including pointers to free memory linked-lists containing references to memory blocks of a size greater than 4K. Branches 210, 215, and 225 serve as head pointers to free memory linked-lists 262, 266, and 272, which include memory blocks of size 1K, 4K, and 64K, respectively. Fast access tree 204 also includes two main branches, 230 and 240. Fast access tree 204 further includes head pointers 250, 255, and 260 to free memory linked-lists 264, 268, and 270, including memory blocks of size 2K, 8K, and 16K, respectively, indicating that these memory block sizes were most frequently accessed by the threads executing in the associated program. General access tree 202 may include additional branches that include head pointers to additional free memory linked-lists. Fast access tree 204, however, only includes head pointers to a specific number of free memory linked-lists, typically no more than eight. A "head pointer" is a pointer to the beginning of a free memory linked-list. Thus, because the fast access tree does not typically grow as large as the general access tree, accesses to the fast access tree are faster than accesses to the general access tree.

Figs. 3A & 3B depict a flow diagram of a smart-alloc function operating in accordance with the principles of the present invention. First, smart-alloc requests access to an amount of memory from a system memory call included as part of an operating system (step 305). If the amount of memory is available (step 310), the system memory call returns to smart-alloc a pointer to the amount of memory requested (step 315). Upon receiving a pointer to a requested amount of memory, smart-alloc divides the memory into blocks of varying sizes and builds a free

LAW OFFICES
FINNEGAN, HENDERSON,
FARABOW, GARRETT,
& DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

memory linked-list for each block size (step 318). The block sizes in the free memory linked-lists begin at 1K and increase in an amount equal to double the size of a previous block. For example, 1K, 2K, 4K, 8K, 16K, etc. Once the free memory linked-lists are created, smart-alloc creates the general access tree which includes head pointers to each of the free memory linked-lists (step 320). Initially, the fast access tree is empty. As a program executes and a series of memory requests are processed, the fast access tree is populated as discussed below.

Upon receiving a memory request from an executing thread (step 325), smart-alloc increments the value of a counter associated with the block size used to satisfy the request (step 330). Thus, if a thread requests 1026 bytes of memory and the free memory linked-lists include 1K and 2K blocks, a reference to the 2K block will be returned to the requesting thread, and the value of the counter associated with the 2K free memory linked-list will be incremented. If this memory request is satisfied from the fast access tree (step 335), processing ends.

Otherwise, if the memory request is satisfied from the general access tree (step 340), then the counter associated with the block size used to satisfy the request is compared with the counter for block sizes pointed to by the fast access tree (step 370). Smart-alloc then determines whether the block size should be added to the free memory linked-list pointed to by the fast access tree step 375 because it includes memory blocks of a size most often requested by the executing program. In this step, smart-alloc compares the value of the counter associated with a block size included in the general access tree with the value of the counter of a block size included in the fast access tree. If the counter value for the block in the general access tree is greater than the counter value for the block in the fast access tree, the pointer from the general access will be replaced with a pointer from the fast access tree. Additionally, the fast access tree pointer that

LAW OFFICES
FINNEGAN, HENDERSON,
FARABOW, GARRETT,
& DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

points to the block size having the lowest counter value is replaced with the pointer from the general access tree (step 380). Changing the pointers in this manner ensures that the fast access tree continually points to the most frequently allocated block sizes, thereby increasing the efficiency of memory allocation. Otherwise, if the counter value associated with the block in the general access tree is not greater than the counter value of any of the free memory linked-lists pointed to by the fast access tree, processing ends.

If a memory request cannot be satisfied from either the general or fast access trees, smart-alloc requests additional memory from a system memory call (step 345). If the additional memory is available (step 350), a pointer to the amount of memory is returned to smart-alloc (step 355), the memory is divided into blocks that are added to the free memory linked-lists pointed to by a pointer from the general access tree (step 360), and processing continues to step 335. Otherwise, if the requested amount of memory is not available, the system memory call returns a null pointer to smart-alloc (step 365), and processing ends.

Fig. 4 depicts a flow diagram of operations performed by smart-alloc when a thread releases a reference to a block of memory. Upon receiving a reference to a block of memory from an executing thread (step 410), smart-alloc adds the reference to the appropriate free memory linked-list (step 430) and processing ends.

LAW OFFICES
FINNEGAN, HENDERSON,
FARABOW, GARRETT,
& DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

Conclusion

Methods and systems operating in accordance with the principles of the present invention optimize the process of allocating memory to threads executing in programs operating in a data processing system by adapting the allocation process according to the behavior of an executing program. This smart-alloc function is a library function that may be linked to any program executing in a data processing system. As discussed above, smart-alloc operates in the user space and does not require accessing the operating system each time memory is allocated or deallocated. These distributive and adaptive features of smart-alloc allow it to minimize the number of accesses to the operating system, a costly and time consuming event, and ultimately yields increased system performance.

Methods and systems consistent with the present invention are applicable to all single and multi-threaded programs written in all computer programming languages, including Fortran 77, Fortran 90, Java, C, and C++. Although maximum increases in efficiency will be realized in a multi-processor computing environment, methods and systems consistent with the present invention may also provide operational benefits in a single or multi-threaded, single processor environment.

Although the foregoing description has been described with reference to a specific implementation, those skilled in the art will know of various changes in form and detail which may be made without departing from the spirit and scope of the present invention as defined in the appended claims and the full scope of their equivalents.

LAW OFFICES
FINNEGAN, HENDERSON,
FARABOW, GARRETT,
& DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

WHAT IS CLAIMED IS:

1. A method in a data processing system for allocating memory by a memory allocation function, comprising the steps defined by the memory allocation function of:

receiving a memory request for a reference to a block of memory;

returning the reference to the block of memory to satisfy the request; and

adjusting an operation of the memory allocation function based on the memory request.

2. The method of claim 1, further including the step of forming a plurality of linked-lists referring to memory blocks of a common size.

3. The method of claim 2, wherein the step of returning includes the step of setting a fast access tree to refer to a first of the plurality of linked-lists.

4. The method of claim 3, further including a step of ensuring that the fast access tree refers to one of the plurality of linked-lists that is most frequently requested.

5. The method of claim 2, wherein the step of returning includes the step of setting a general access tree to refer to a second of the plurality of linked-lists.

LAW OFFICES
FINNEGAN, HENDERSON,
FARABOW, GARRETT,
& DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

6. A method in a data processing system for providing access to a memory that includes an operating system with a system memory call, the memory further including a program which includes a memory access function, comprising the steps performed by the memory access function of:

requesting access to a portion of memory via the system memory call;

receiving from the system memory call a pointer to the portion of memory;

dividing the portion of memory into memory blocks, a plurality of the memory blocks being of different sizes;

forming a plurality of linked-lists, each linked-list referring to memory blocks of a common size, each linked-list having an associated counter;

setting a fast access tree to refer to a first of the plurality of linked-lists;

setting a general access tree to refer to a second of the plurality of linked-lists;

receiving a memory request;

determining which among the plurality of linked-lists contains a memory block that will satisfy the memory request;

incrementing the counter associated with the determined linked-list;

returning a reference to the memory block on the determined linked-list;

comparing the counters of the plurality of linked-lists to identify a predetermined number of linked-lists with a largest counter; and

ensuring that the fast access tree is set to refer to the identified linked-lists with the largest counter.

LAW OFFICES
FINNEGAN, HENDERSON,
FARABOW, GARRETT,
& DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

7. A system for allocating memory, comprising:

means for receiving a memory request for a reference to a block of memory;

means for returning the reference to the block of memory to satisfy the request; and

means for adjusting an operation of a memory access function based on the memory request.

LAW OFFICES
FINNEGAN, HENDERSON,
FARABOW, GARRETT,
& DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

8. A data processing system for providing access to memory, comprising:

a memory including:

a program including a memory access function that provides access to memory and that adjusts its operation according to a memory request for a reference to a block of memory; and

a processor for executing the program.

9. The data processing system of claim 8, further including an operating system with a system memory function, and wherein the memory access function provides access to memory by utilizing the system memory function.

10. The data processing system of claim 8, wherein the memory access function includes a plurality of linked-lists referred to by a fast access tree.

11. The data processing system of claim 10, wherein the fast access tree refers to one of the plurality of linked-lists that is most frequently accessed.

12. The data processing system of claim 10, wherein a most frequently accessed memory block size is included in the fast access tree.

LAW OFFICES
FINNEGAN, HENDERSON,
FARABOW, GARRETT,
& DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

13. The data processing system of claim 8, wherein the memory access function includes a plurality of linked-lists referred to by a general access tree.

14. The data processing system of claim 13, wherein a least frequently accessed memory block size is included in the general access tree.

15. The data processing system of claim 8, further including a plurality of linked-lists, each linked-list referring to memory blocks of a common size.

16. The data processing system of claim 15, wherein each of the plurality of linked-lists has an associated counter indicating a number of times that the associated linked-list has been accessed.

LAW OFFICES
FINNEGAN, HENDERSON,
FARABOW, GARRETT,
& DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

17. A computer-readable medium including instructions for performing a method for allocating memory by a memory allocation function, the method comprising the steps performed by the memory allocation function of:

receiving a memory request for a reference to a block of memory;

returning the reference to the block of memory to satisfy the request; and

adjusting an operation of the memory allocation function based on the memory request.

18. The computer-readable medium of claim 15, further including instructions for forming a plurality of linked-lists referring to memory blocks of a common size.

19. The computer-readable medium of claim 18, wherein the instructions for returning include instructions for setting a fast access tree to refer to a first of the plurality of linked-lists.

20. The computer-readable medium of claim 19, further including instructions for inserting a most frequently accessed memory block size into the fast access tree.

21. The computer-readable medium of claim 19, further including instructions for ensuring that the fast access tree refers to one of the plurality of linked-lists that is most frequently requested.

LAW OFFICES
FINNEGAN, HENDERSON,
FARABOW, GARRETT,
& DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

22. The computer-readable medium of claim 18, wherein the instructions for returning include instructions for setting a general access tree to refer to a second of the plurality of linked-lists.

23. The computer-readable medium of claim 22, further including instructions for inserting a least frequently accessed memory block size into the general access tree.

LAW OFFICES
FINNEGAN, HENDERSON,
FARABOW, GARRETT,
& DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

ABSTRACT

This functions maintains two trees: a fast access tree referring to memory blocks of a size most often requested, and a general access tree referring to memory blocks of a size less often requested. After satisfying a request for a memory block, the function adjusts the trees to ensure that the fast access tree refers to memory blocks of the size most often requested. By providing such functionality, the function improves its performance over time through self-adaptation.

LAW OFFICES
FINNEGAN, HENDERSON,
FARABOW, GARRETT,
& DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

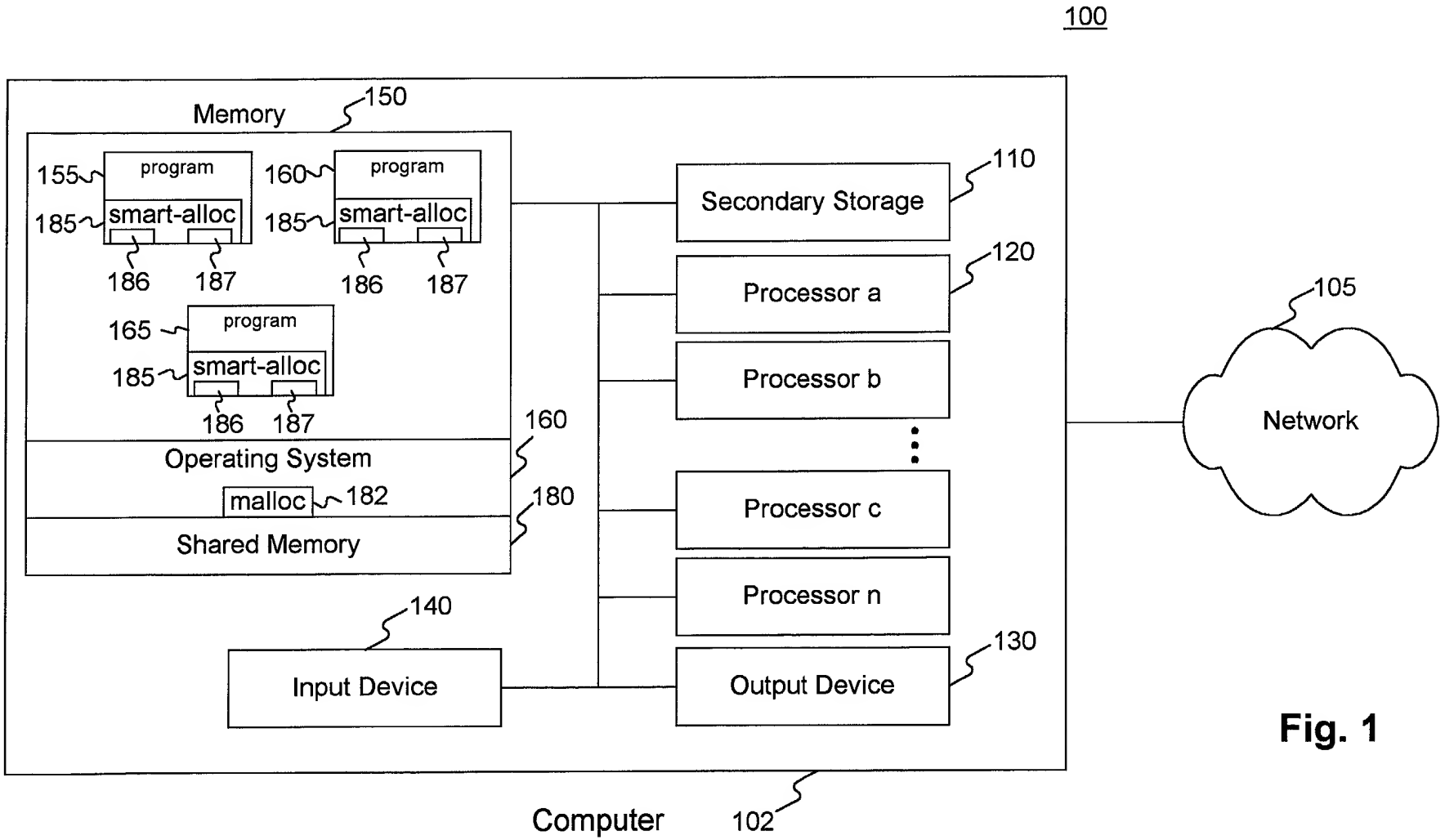

Fig. 1

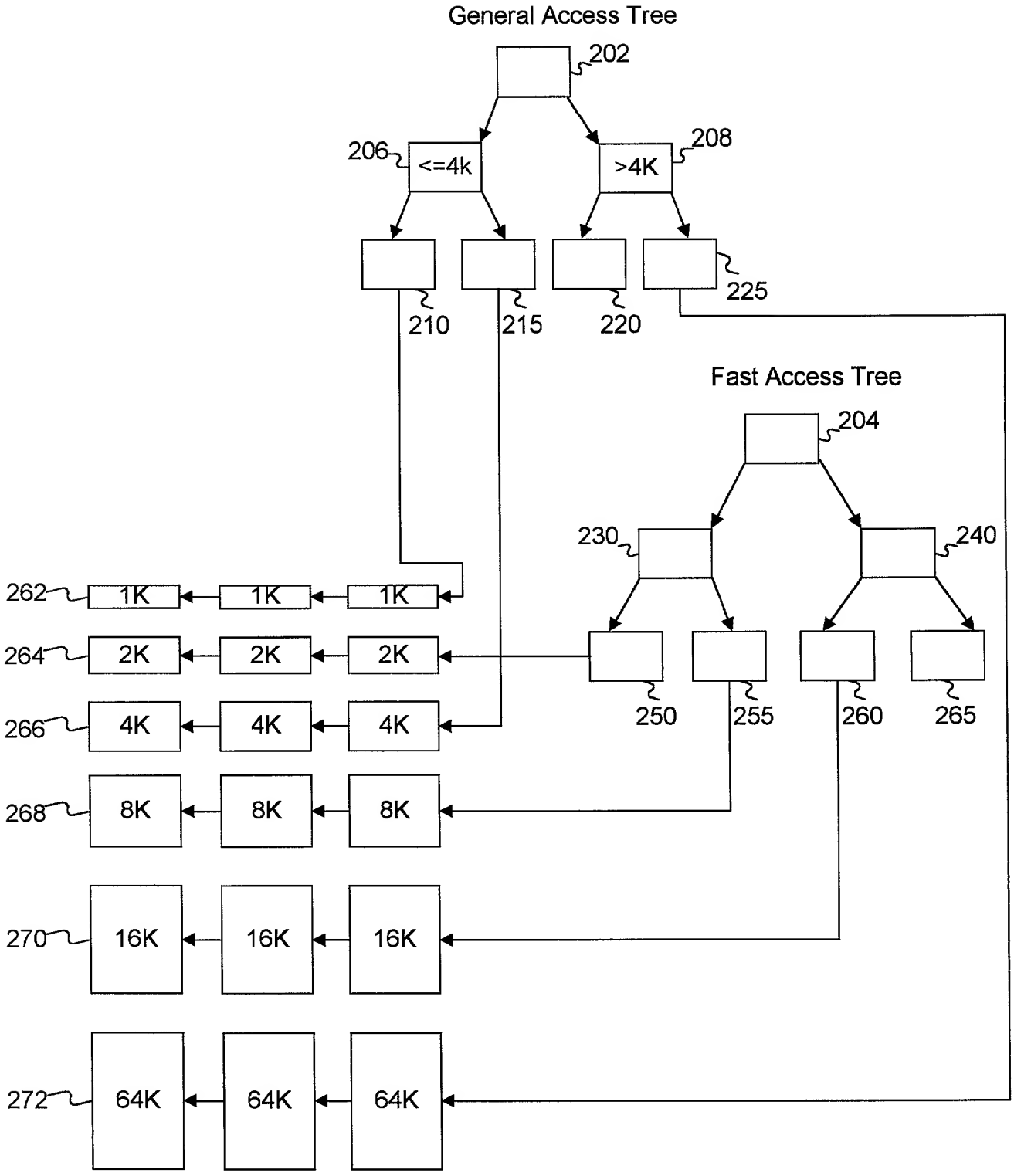
General Access Tree
202
206
<=4k
208
>4K
210
215
220
225
Fast Access Tree
204
230
240
250
255
260
265
262
1K
1K
1K
264
2K
2K
2K
266
4K
4K
4K
268
8K
8K
8K
270
16K
16K
16K
272
64K
64K
64K

Fig. 2

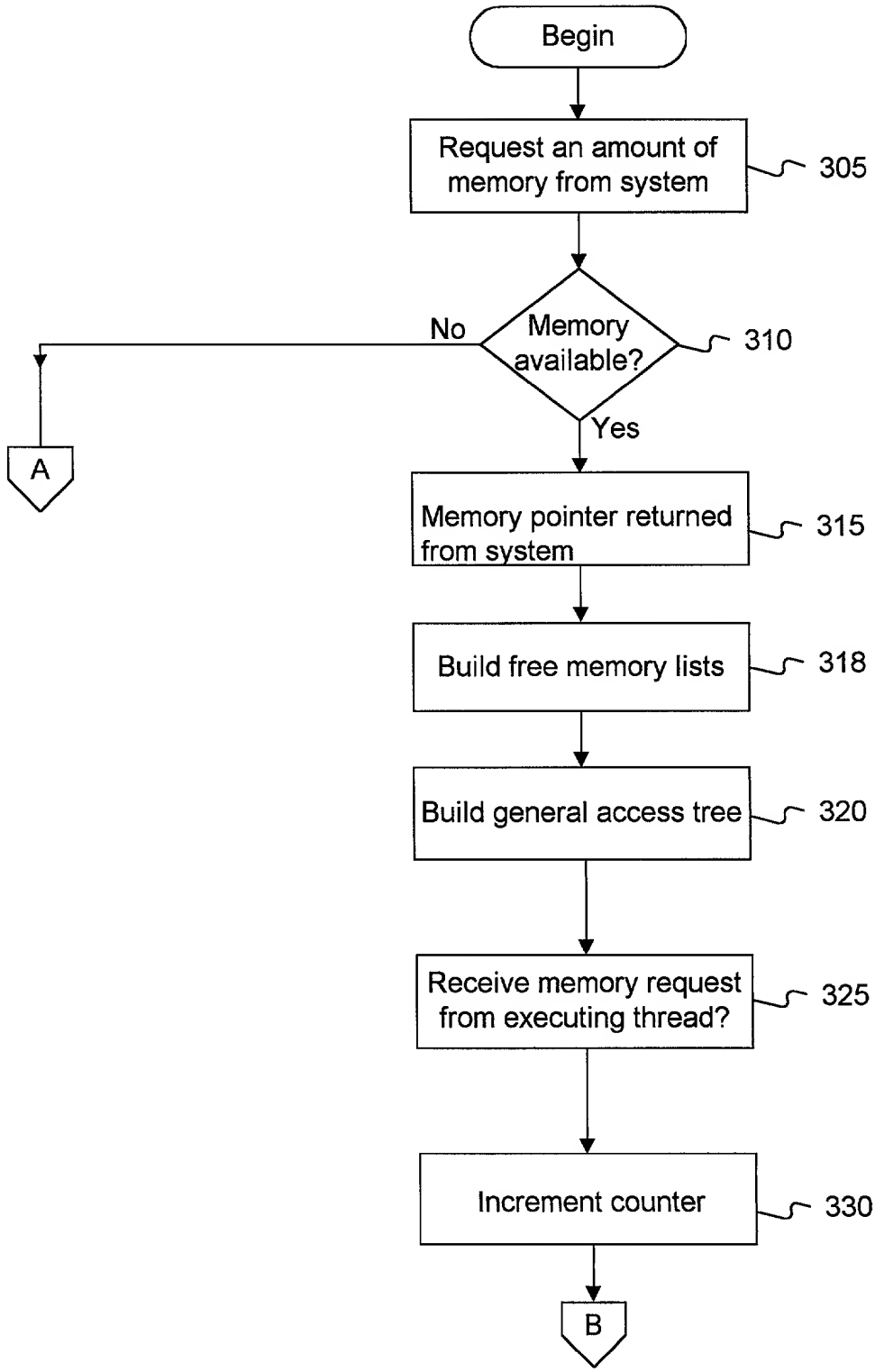
Begin
Request an amount of memory from system
305
Memory available?
310
No
Yes
A
Memory pointer returned from system
315
Build free memory lists
318
Build general access tree
320
Receive memory request from executing thread?
325
Increment counter
330
B

Fig. 3A

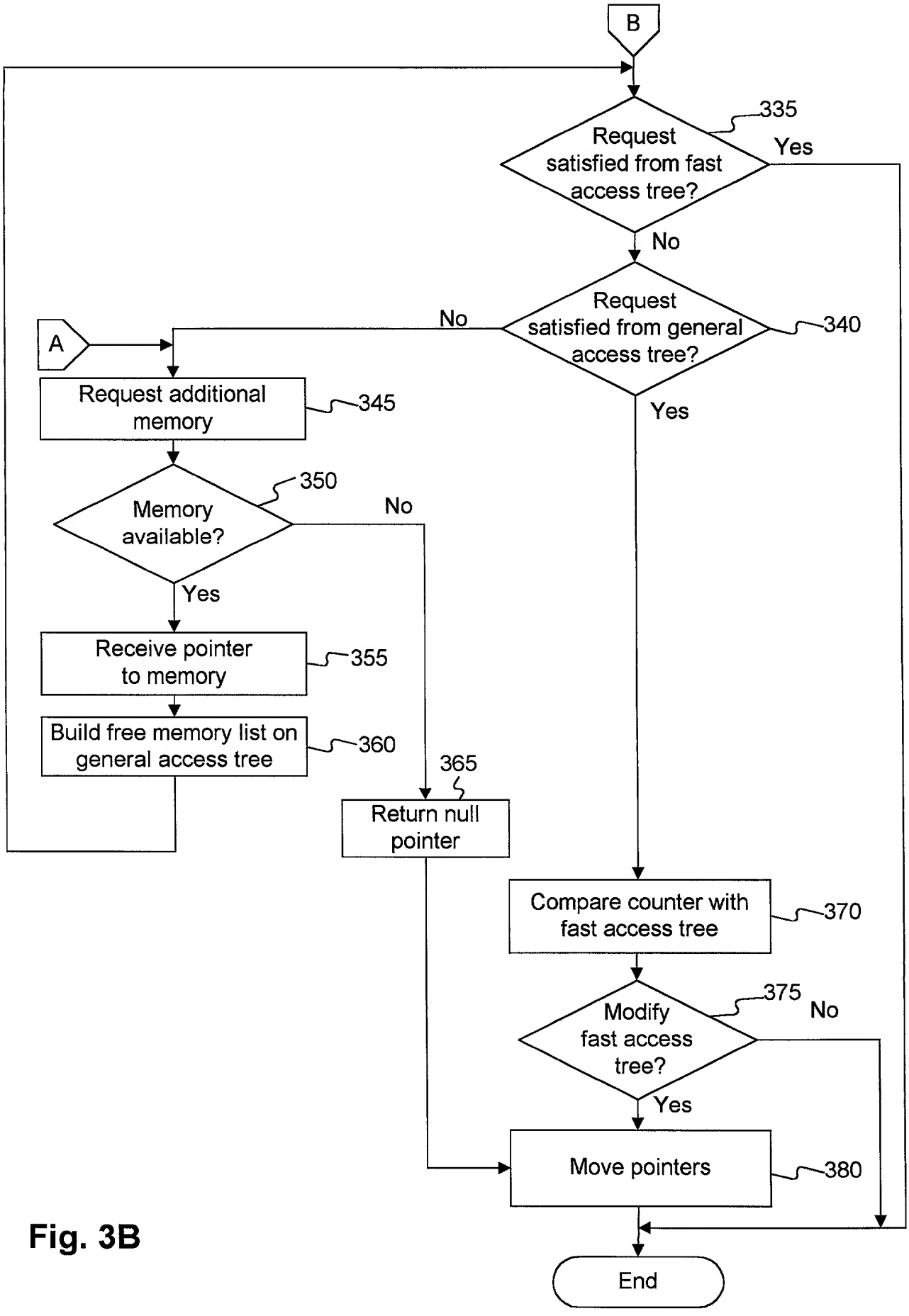
B
Request satisfied from fast access tree?
335
Yes
No
Request satisfied from general access tree?
340
No
A
Request additional memory
345
Yes
Memory available?
350
No
Yes
Receive pointer to memory
355
Build free memory list on general access tree
360
Return null pointer
365
Compare counter with fast access tree
370
Modify fast access tree?
375
No
Yes
Move pointers
380
End

Fig. 3B

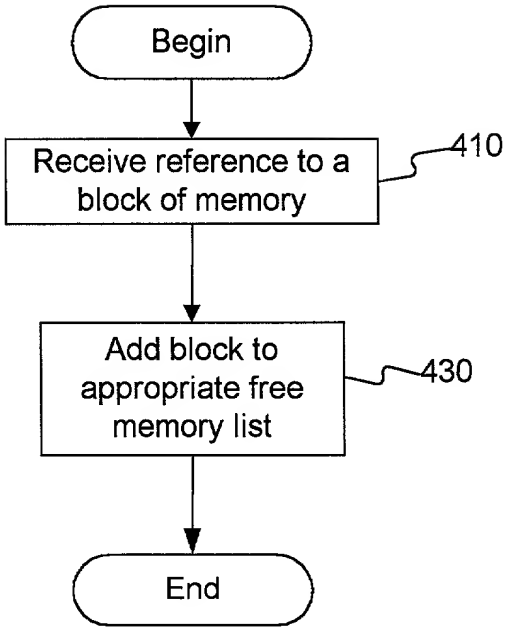
Begin
Receive reference to a block of memory
410
Add block to appropriate free memory list
430
End

Fig. 4

Attorney Docket No. 06502.0210
Sun Reference No. P3689/AES

DECLARATION AND POWER OF ATTORNEY

As a below named inventor, I hereby declare that: my residence, post office address and citizenship are as stated below next to my name; I believe I am the original, first, and sole inventor (if only one name is listed below) or an original, first, and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled: **ADAPTIVE MEMORY ALLOCATION**

_____ the specification of which ☒ is attached and/or ☐ was filed on _____ as United States Application Serial No. _____ or PCT International Application No. _____ and was amended on _____ (if applicable).

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above. I acknowledge the duty to disclose information which is material to patentability as defined in 37 CFR § 1.56.

I hereby claim foreign priority benefits under 35 U.S.C. § 119(a)-(d) or § 365(b) of any foreign application(s) for patent or inventor's certificate or § 365(a) of any PCT International application(s) designating at least one country other than the United States, listed below and have also identified below, any foreign application(s) for patent or inventor's certificate, or any PCT International application(s) having a filing date before that of the application(s) of which priority is claimed:

Country	Application Number	Date of Filing	Priority Claimed Under 35 U.S.C.
			☐ YES ☐ NO
			☐ YES ☐ NO

I hereby claim the benefit under 35 U.S.C. § 119(e) of any United States provisional application(s) listed below:

Application Number	Date of Filing

I hereby claim the benefit under 35 U.S.C. § 120 of any United States application(s) or § 365(c) of any PCT International application(s) designating the United States, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT International application(s) in the manner provided by the first paragraph of 35 U.S.C. § 112, I acknowledge the duty to disclose information which is material to patentability as defined in 37 CFR § 1.56 which became available between the filing date of the prior application(s) and the national or PCT International filing date of this application:

Application Number	Date of Filing	Status (Patented, Pending, Abandoned)

I hereby appoint the following attorney and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith. FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L.L.P., Douglas B. Henderson, Reg. No. 20,291; Ford F. Farabow, Jr., Reg. No. 20,630; Arthur S. Garrett, Reg. No. 20,338; Donald R. Dunner, Reg. No. 19,073; Brian G. Brunsvold, Reg. No. 22,593; Tipton D. Jennings, IV, Reg. No. 20,645; Jerry D. Voight, Reg. No. 23,020; Laurence R. Hefter, Reg. No. 20,827; Kenneth E. Payne, Reg. No. 23,098; Herbert H. Mintz, Reg. No. 26,691; C. Larry O'Rourke, Reg. No. 26,014; Albert J. Santorelli, Reg. No. 22,610; Michael C. Elmer, Reg. No. 25,857; Richard H. Smith, Reg. No. 20,609; Stephen L. Peterson, Reg. No. 26,325; John M. Romary, Reg. No. 26,331; Bruce C. Zotter, Reg. No. 27,680; Dennis P. O'Reilley, Reg. No. 27,932; Allen M. Sokal, Reg. No. 26,695; Robert D. Balefsky, Reg. No. 25,387; Richard L. Stroup, Reg. No. 28,478; David W. Hill, Reg. No. 28,220; Thomas L. Irving, Reg. No. 28,619; Charles E. Lipsey, Reg. No. 28,165; Thomas W. Winland, Reg. No. 27,605; Basil J. Lewris, Reg. No. 28,818; Martin I. Fuchs, Reg. No. 28,508; E. Robert Yoches, Reg. No. 30,120; Barry W. Graham, Reg. No. 29,924; Susan Haberman Griffen, Reg. No. 30,907; Richard B. Racine, Reg. No. 30,415; Thomas H. Jenkins, Reg. No. 30,857; Robert E. Converse, Jr., Reg. No. 27,432; Clair X. Mullen, Jr., Reg. No. 20,348; Christopher P. Foley, Reg. No. 31,354; John C. Paul, Reg. No. 30,413; Roger D. Taylor, Reg. No. 28,992; David M. Kelly, Reg. No. 30,953; Kenneth J. Meyers, Reg. No. 25,146; Carol P. Einaudi, Reg. No. 32,220; Walter Y. Boyd, Jr., Reg. No. 31,738; Steven M. Anzalone, Reg. No. 32,095; Jean B. Fordis, Reg. No. 32,984; Barbara C. McCurdy, Reg. No. 32,120; James K. Hammond, Reg. No. 31,964; Richard V. Burgujian, Reg. No. 31,744; J. Michael Jakes, Reg. No. 32,824; Thomas W. Banks, Reg. No. 32,719; Christopher P. Isaac, Reg. No. 32,616; Bryan C. Diner, Reg. No. 32,409; M. Paul Barker, Reg. No. 32,013; Andrew Chanho Sonu, Reg. No. 33,457; David S. Forman, Reg. No. 33,694; Vincent P. Kovalick, Reg. No. 32,867; James W. Edmondson, Reg. No. 33,871; Michael R. McGurk, Reg. No. 32,045; Joann M. Neth, Reg. No. 36,363; Gerson S. Panitch, Reg. No. 33,751; Cheri M. Taylor, Reg. No. 33,216; Charles E. Van Horn, Reg. No. 40,266; Linda A. Wadler, Reg. No. 33,218; Jeffrey A. Berkowitz, Reg. No. 36,743; Michael R. Kelly, Reg. No. 33,921; James B. Monroe, Reg. No. 33,971; Doris Johnson Hines, Reg. No. 34,629; Allen R. Jensen, Reg. No. 28,224; Lori Ann Johnson, Reg. No. 34,498; and David A. Manspeizer, Reg. No. 37,540; Kenneth Olsen, Reg. No. 26,493; Timothy J. Crean, Reg. No. 37,116; Joseph T. FitzGerald, Reg. No. 33,881; Robert S. Hauser, Reg. No. 37,847; Alexander E. Silverman, Reg. No. 37,940; Christine S. Lam, Reg. No. 37,489; Anirma Rakshpal Gupta, Reg. No. 38,275; Sean P. Lewis, Reg. No. 42,798; Michael J. Schallop, Reg. No. 44,319; Bernice B. Chen, Reg. No. 42,403; Kenta Suzue, Reg. No. 45,145; Noreen A. Krall, Reg. No. 39,734 and Richard J. Lutton, Jr., Reg. No. 39,756 . Please address all correspondence to FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L.L.P. 1300 I Street, N.W., Washington, D.C. 20005, Telephone No. (202) 408-4000.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

Listing of Inventors Continued on Page 2 hereof. ☐ Yes ☐ No

Listing of Inventors Continued From Page 1 hereof.

Attorney Docket No. 06502.0210
Sun Reference No. P3689/AES

Full Name of First Inventor Paul HINKER	Inventor's Signature Paul Hinker	Date 2/11/00
Residence 2420 Atwood Street, Longmont CO 80501		Citizenship USA
Post Office Address		
Full Name of Second Inventor Bradley LEWIS	Inventor's Signature Bradley Lewis	Date 2/11/00
Residence 1220 West 12th Avenue Broomfield CO 80020		Citizenship USA
Post Office Address		
Full Name of Third Inventor Michael BOUCHER	Inventor's Signature Michael Boucher	Date 2/11/2000
Residence 1769 Casey Court Lafayette, CO 80026		Citizenship USA
Post Office Address		

Listing of Inventors Continued on Page 2 hereof. ☐ Yes ☐ No